

Contents

1 Routine/Function Prologues	2
1.1 Fortran: Module Interface def_ipMod.F90 (Source File: def_ipMod.F90) . . .	2
1.1.1 allocate_ip (Source File: def_ipMod.F90)	2
1.1.2 def_ip_input (Source File: def_ipMod.F90)	3

1 Routine/Function Prologues

1.1 Fortran: Module Interface def_ipMod.F90 (Source File: def_ipMod.F90)

This module contains routines that precomputes weights and other parameters required for spatial interpolation of model forcing

REVISION HISTORY:

14Nov02 Sujay Kumar Initial Specification

INTERFACE:

```
module def_ipMod
  implicit none
```

ARGUMENTS:

```
real, allocatable :: rlat0(:)
real, allocatable :: rlon0(:)
integer, allocatable :: n110(:)
integer, allocatable :: n120(:)
integer, allocatable :: n210(:)
integer, allocatable :: n220(:)
real, allocatable :: w110(:,w120(:))
real, allocatable :: w210(:,w220(:))
integer :: mi, mo, nn
integer :: nb3,nb4
```

1.1.1 allocate_ip (Source File: def_ipMod.F90)

Allocates memory for interpolation of model forcing data (GEOS and GDAS)

INTERFACE:

```
subroutine allocate_ip(N)
```

ARGUMENTS:

```
integer :: N
```

CONTENTS:

```
allocate(rlat0(n))
allocate(rlon0(n))
allocate(n110(n))
allocate(n120(n))
allocate(n210(n))
allocate(n220(n))
allocate(w110(n))
```

```

allocate(w120(n))
allocate(w210(n))
allocate(w220(n))
mo = n
nn = n

w110 = 0.0
w120 = 0.0
w210 = 0.0
w220 = 0.0

```

1.1.2 def_ip_input (Source File: def_ipMod.F90)

Calculates spatial variables required for interpolation of GEOS/GDAS model forcing

INTERFACE:

```
subroutine def_ip_input (kgdsi)
```

USES:

```

use spmdMod
use lisdrv_module, only:lis
#if ( defined OPENDAP )
    use opendap_module
#endif
!INPUT ARGUMENTS:
integer :: kgdsi(200)

```

CONTENTS:

```

#if ( ! defined OPENDAP )
    integer :: nroffset = 0
#endif
    kgdso = lis%d%kgds
#if ( defined OPENDAP )
    mo = parm_nc*parm_nr
#else
    mo = lis%d%lnc*lis%d%lnr
#endif
!-----
! Calls the routines to decode the grid description and
! calculates the weights and neighbor information to perform
! spatial interpolation. This routine eliminates the need to
! compute these weights repeatedly during interpolation.
!-----
if(kgdso(1).ge.0) then
    call gdswiz(kgdso, 0,mo,fill,xpts,ypts,rlon0,rlat0,nn,0)

```

```

        endif
        call gdswiz(kgdsi,-1,nn,fill,xpts,ypts,rlon0,rlat0,nv,0)
        do n=1,nn
            xi=xpts(n)
            yi=ypts(n)
            if(xi.ne.fill.and.yi.ne.fill) then
                i1=xi
                i2=i1+1
                j1=yi
                j2=j1+1
                xf=xi-i1
                yf=yi-j1
                n110(n)=ijkgs(i1,j1,kgdsi)
                n210(n)=ijkgs(i2,j1,kgdsi)
                n120(n)=ijkgs(i1,j2,kgdsi)
                n220(n)=ijkgs(i2,j2,kgdsi)
                if(min(n110(n),n210(n),n120(n),n220(n)).gt.0) then
                    w110(n)=(1-xf)*(1-yf)
                    w210(n)=xf*(1-yf)
                    w120(n)=(1-xf)*yf
                    w220(n)=xf*yf
                else
                    n110(n)=0
                    n210(n)=0
                    n120(n)=0
                    n220(n)=0
                endif
            else
                n110(n)=0
                n210(n)=0
                n120(n)=0
                n220(n)=0
            endif
        enddo
#if ( defined OPENDAP )
!     mi = geos_nc*geos_nr
#else
!     mi = lis%f%ncold*lis%f%nrold
#endif

```